Computer Vision homework 5

Luis Valle

November 28th 2016

0.1 Q1.1.1

ReLU is preferred because there is less likelihood of vanishing gradient and there is sparsity. ReLU is not as computationally expensive as sigmoids and tanh, where exponentials are required. The convergence of stochastic gradient descent is faster with this approach than previously mentioned.

0.2 section 1.1.2

Linear functions of linear functions result in another linear function. E.g.

$$g_1 = W_1 X + b_1$$

$$g_2 = W_2 g_1 + b_2$$

$$g_2 = W_2(W_1 X + b_1) + b_2$$

$$g_2 = W_2 W_1 X + W_2 b_1 + b_2$$

We can replace this last line with $W_3 = W_2 W_1$ and $b_3 = W_2 b_1 + b_2$, which gives us:

$$g_2 = W_3 X + b_3$$

0.3 Q2.1.1

It is not advisable to initialize a network with all zeros, all ones, or constant values because the weights need to be asymmetric. This is in order for the different gradients not to undergo the same gradient computation, which would cause the network to get stuck.

0.4 Q2.1.3

I chose to generate a normal distribution and divide by the square root of the current layer. I chose to do this in order to get different weights across the

board and to differentiate the value of the distribution depending on layer for more variety.

0.5 Q2.4.1

Stochastic gradient descent performs well when dealing with a lot of data where there is repetitive data. It is harder than batch gradient descent to get stuck in local minima because each iteration will not follow the same path all the time as in deterministic approaches. However, stochastic gradient descent could kick itself out of the minima when close to convergence and the learning rate is not small enough. It also converges to the minima slower than batch gradient descent.

Batch gradient descent is good when the data has a smooth pattern. It's rate of convergence is a lot faster than stochastic gradient descent. Weights are updated in a single step. However, it does not perform as well with noisy data and can get stuck in loca minima a lot easier than stochastic gradient descent.

In short, batch gradient descent is faster in terms of number of epochs, but stochastic gradient descent is faster in terms of iterations.

0.6 Q3.1.2

For learning rate of 0.01:

Train Accuracy = 0.9749
Train Loss = 1.061
Valid Accuracy = 0.8615
Valid Loss = 1.109

For learning rate of 0.001:

Train Accuracy = 0.6945
Train Loss = 1.247
Valid Accuracy = 0.6708
Valid Loss = 1.251

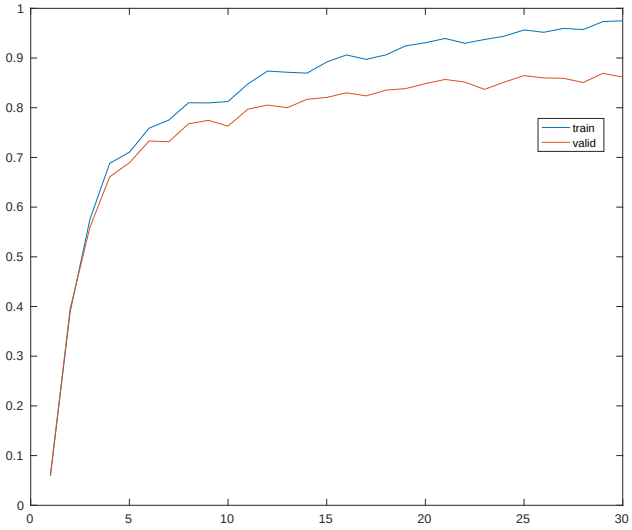

Figure 1: Accuracy with learning Rate 0.01

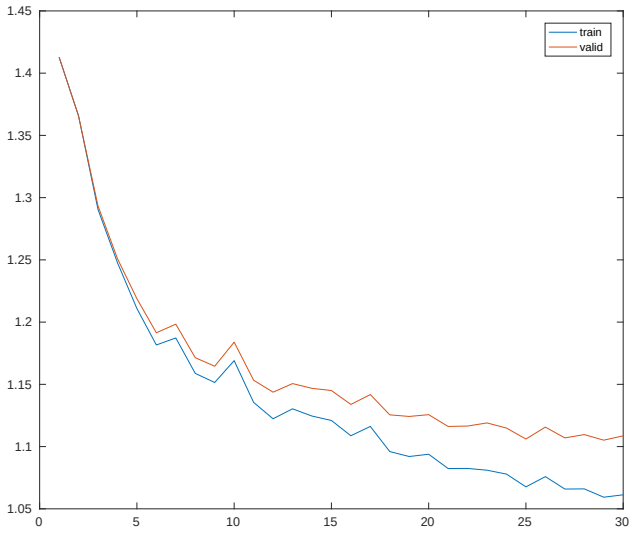

Figure 2: Loss with learning Rate 0.01

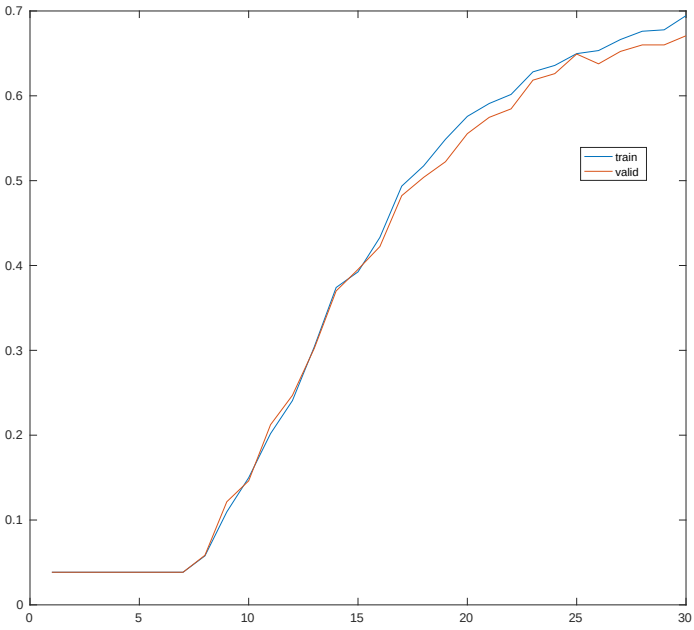

Figure 3: Accuracy with learning Rate 0.001

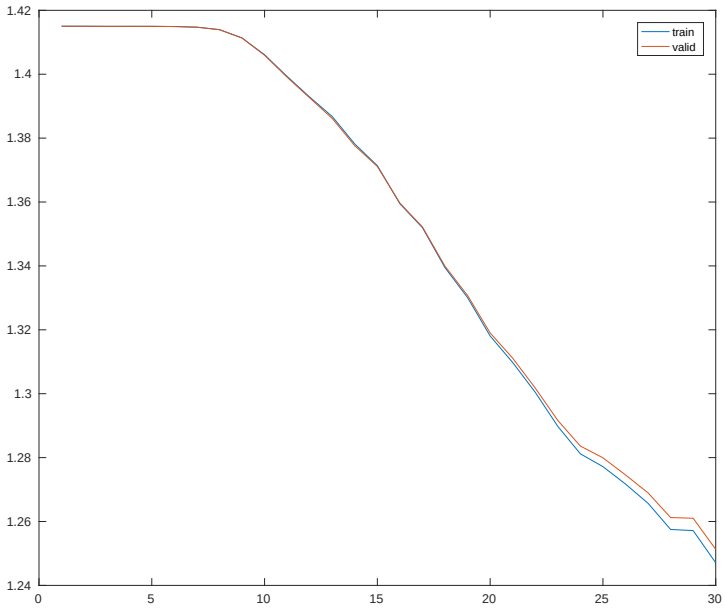

Figure 4: Loss with learning Rate 0.001

0.7 Q3.1.3

There exists patterns in the trained weights. They look like common filters used for image processing like gaussians, derivative of gaussians, horizontal, and vertical filters.

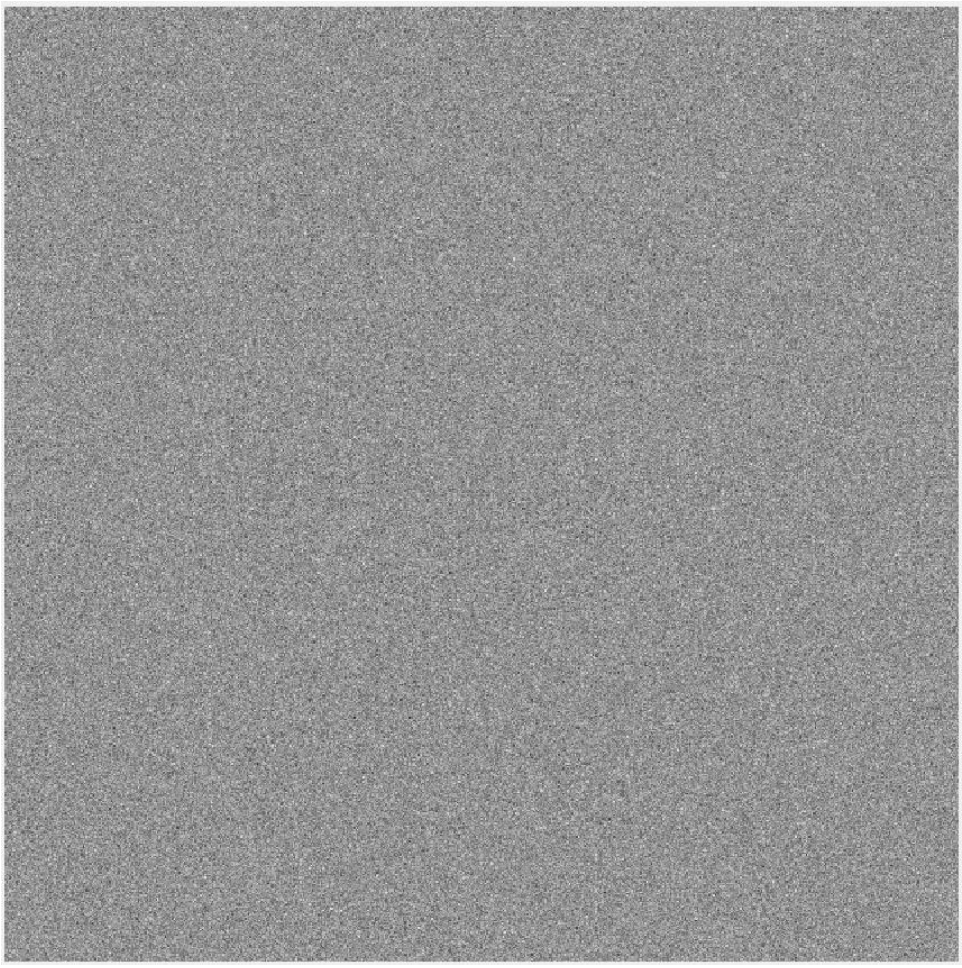

Figure 5: Initialized Weights for nist26

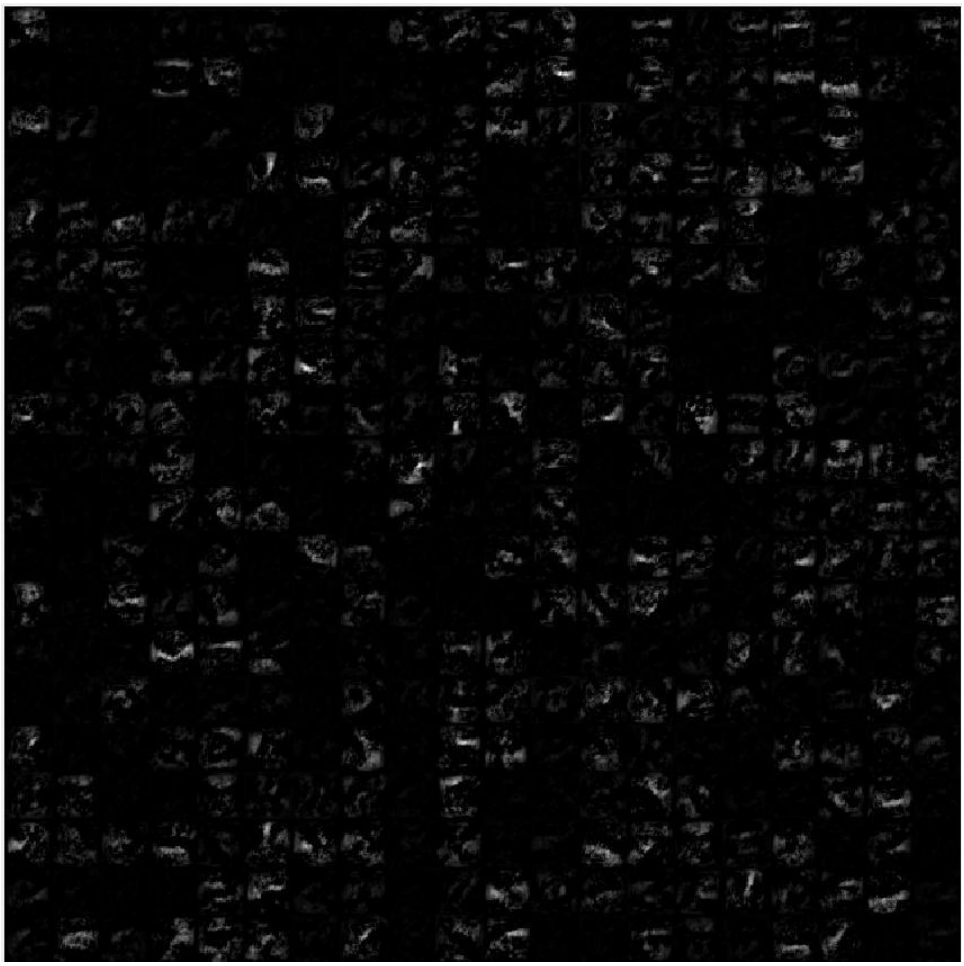

Figure 6: Trained Weights for nist26

0.8 Q3.1.4

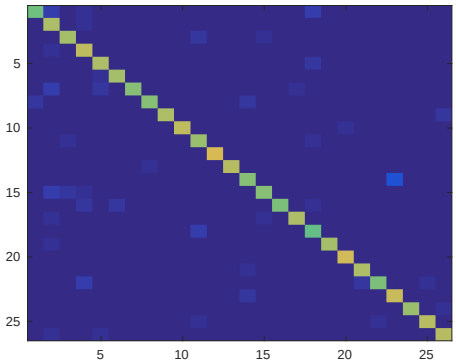

Figure 7: Confusion matrix of nist26. W was replaced with N 7 times. There were a couple letters that were confused 4 times. One of them was R being replaced with K. Both of these cases make sense because in both cases characters are similar to the second character.

0.9 Q3.2.1

Train Accuracy = 0.7182
Train Loss = 1.346
Valid Accuracy = 0.675
Valid Loss = 1.357

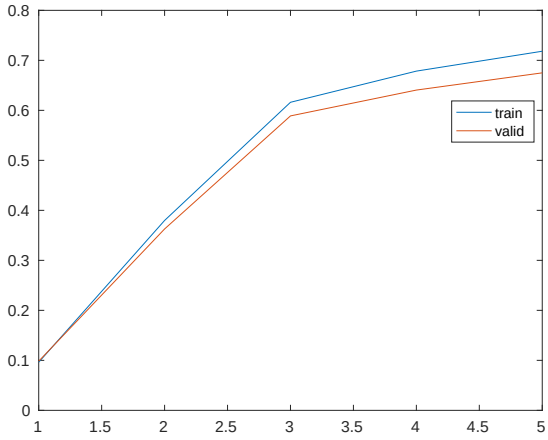

Figure 8: Accuracy for nist36 set

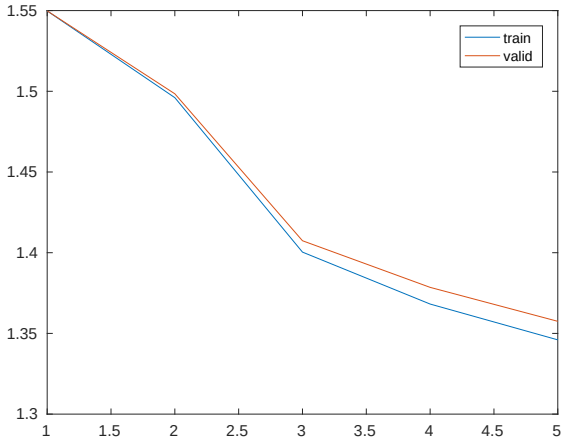

Figure 9: Loss for nist36 set

0.10 Q3.2.2

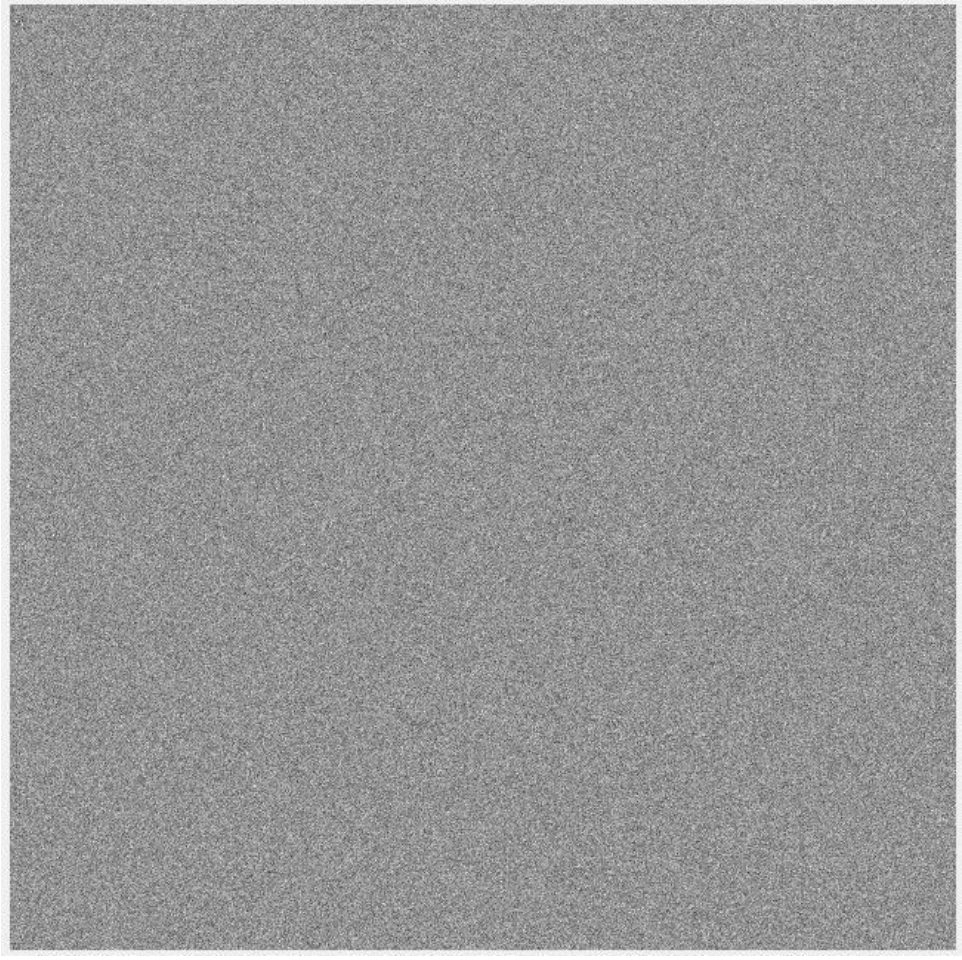

Figure 10: Initialized Weights for nist36

Figure 11: Trained Weights for nist36

0.11 Q3.2.3

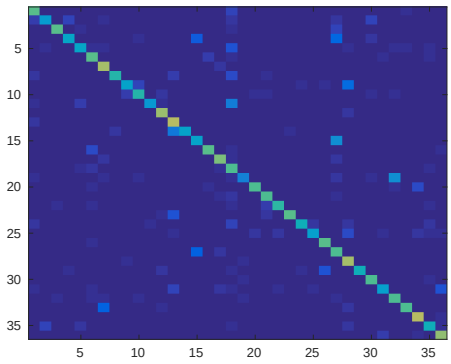

Figure 12: Confusion Matrix of nist36. Two classes were confused 19 times. 0 was replaced with O and 5 was confused with S. Again, just like in nist26, the characters are very similar, so it is expected for them to be confused.

0.12 Q4.1

A couple assumptions are that there will only be characters in the image, that each character will be completely connected and legible. For this particular data set, we assume that we feed in print letters and not cursive.

The following are examples where the classifier could fail due to disconnectedness or noise from other characters.

Figure 13: A 'T' which shows disconnection and noise from another letter.

Figure 14: An 'E' that shows disconnection and could be confused.

0.13 Q4.3

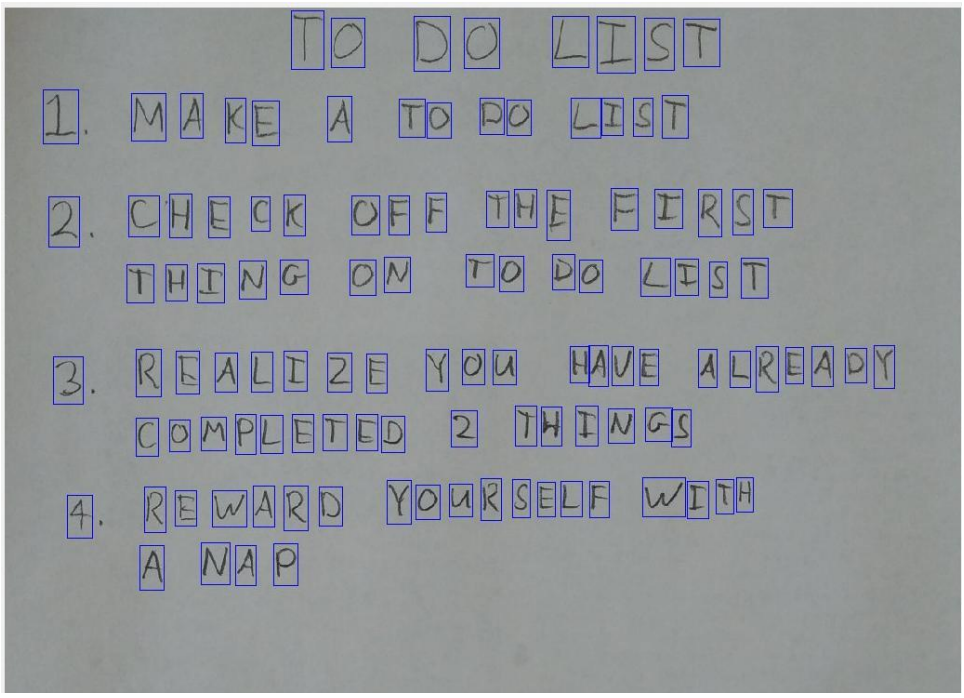

Figure 15: First Image

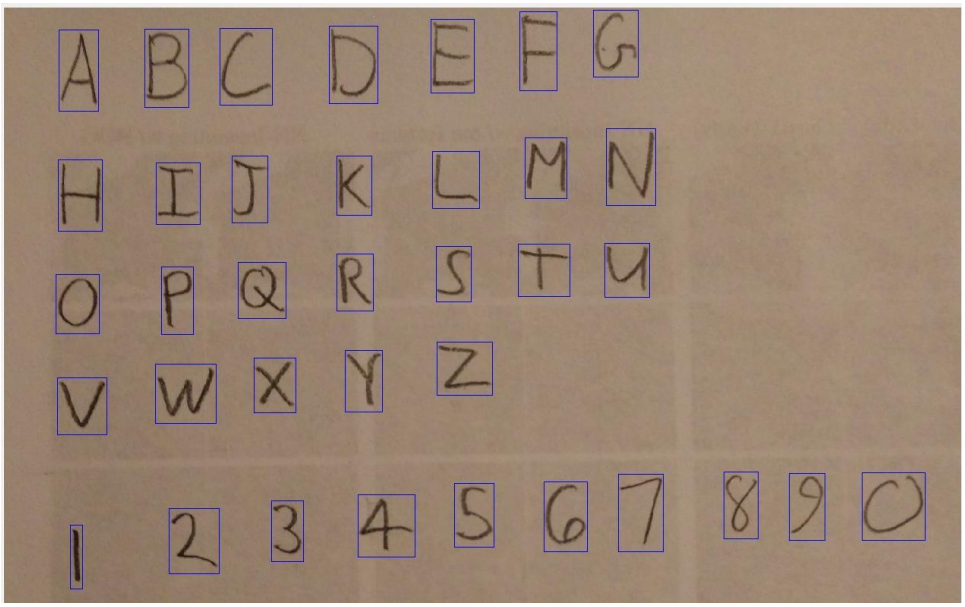

Figure 16: Second Image

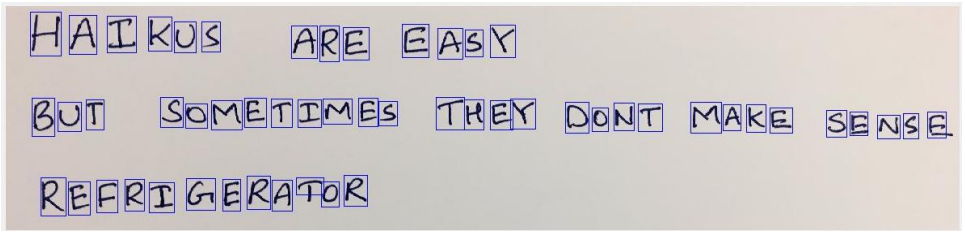

Figure 17: Third Image

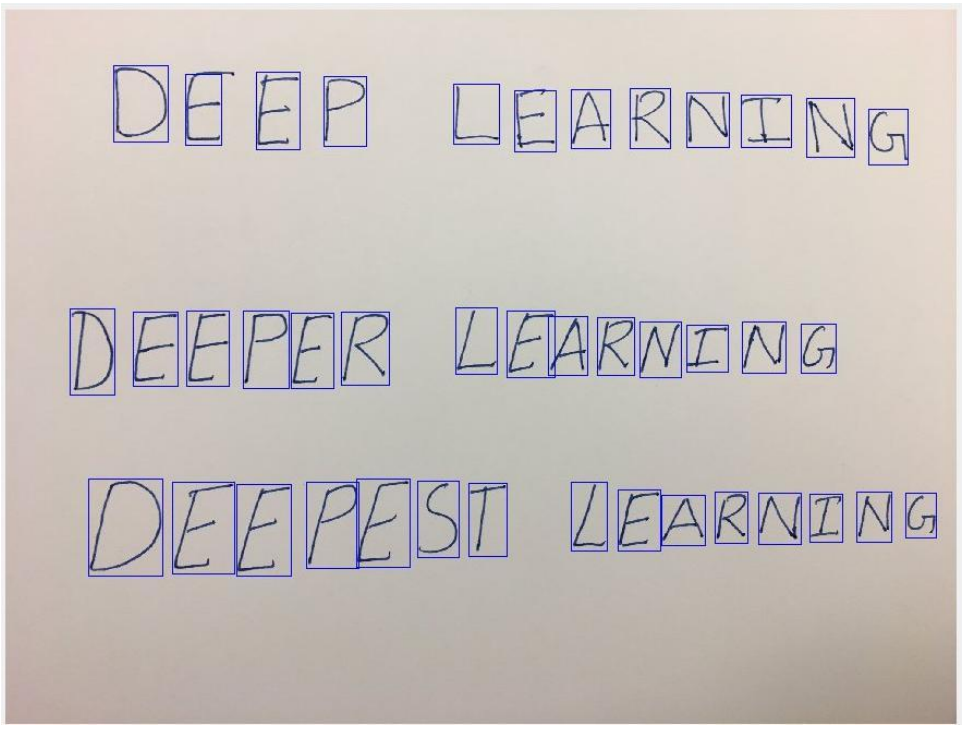

Figure 18: Fourth Image

0.14 Q4.4

First Image:
TQ CJ LIST
L MAKE A TQ QQ LIST
2 CMKCK QFK 7ME FZMST TMIMQ
QM TQ QQ CXST
3 KKALIZK YQM MAVK AWKEAQY
QQMPLETQD Z 7MXMGS
9 KFWARQ YQMKSELK WZTM 9 MAM

Second Image:
A Y L J G F G
M I J K L M W
Q K Q K S T W
V W X Y Z
8 Z 3 G S G Y X Y J

Third Image:
MAIWWS MKE BABY

BWT SQMBTIMQS TMBY DQWT MAKE BGMGG
RBFRIGBKMTQK

Fourth Image:
YYYY YYAKYIYY
YYYYYY LYAKYIMG
YYYYYYY 1EPKMIMG